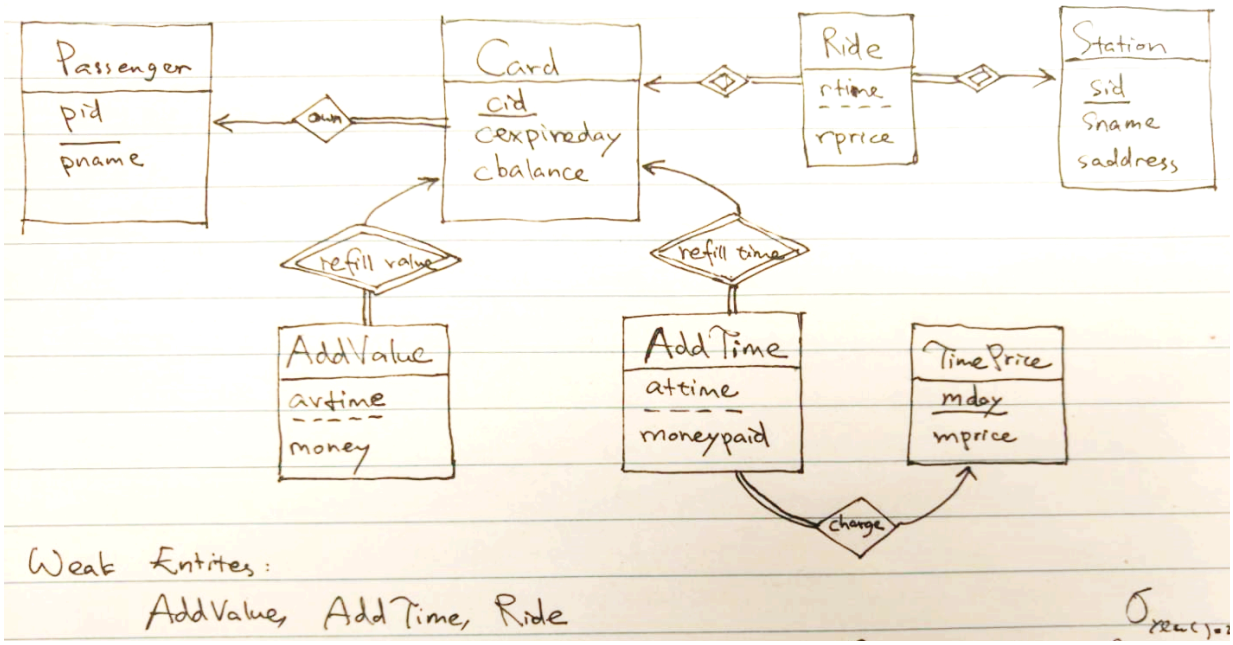
Problem Set #3 Solution

Problem 1

(a)



(b) Answer:

- Card.pid is a foreign key referencing Passenger.pid
- AddTime.cid, AddValue.cid and Ride.cid are foreign keys referencing Card.cid
- AddValue.mday is a foreign key referencing TimePrice.mday
- Ride.sid is a foreign key referencing Station.sid

(c) Answer:

Passenger (<u>pid</u>, pname);
Ownership (<u>pid</u>, <u>cid</u>);
Card (<u>cid</u>, cexpireday, cbalance);
TimePrice (<u>mday</u>, mprice);
Station (<u>sid</u>, sname, saddress);
AddTime (<u>cid</u>, <u>attime</u>, mday, moneypaid);
AddValue (<u>cid</u>, <u>avtime</u>, money);
Ride (<u>cid</u>, <u>sid</u>, <u>rtime</u>, rprice);

(d) Answer:

Passenger (<u>pid</u>, pname);
Card (<u>cid</u>, pid, cexpireday, cbalance);

TimePrice (mday, mprice);
Station (sid, sname, saddress);
AddTime (cid, attime, mday, moneypaid);
AddValue (cid, avtime, money);
Pricetable (departuresid, arrivalsid, pprice)
Ride (cid, departuresid, arrivalsid, departuretime, arrivaltime);

Pricetable.departuresid and Pricetable.arrivalsid are foreign keys referencing Station.sid;
Ride.(departuresid, arrivalsid) is a foreign key referencing Pricetable. (departuresid, arrivalsid).

(e) -------

(f)

(i)

```
select p.pid, count(t.cid)
from Passenger p left outer join (
        select pid, cid, sid
        from Card natural join Ride
        where date(rtime) = '2019-12-25') as t on p.pid = t.pid
group by p.pid
```

pid	count(t.cid)
1	0
2	0
3	0
4	0
5	0
6	2
7	10

(ii)

```
create view v1 as
        select p.pid, sum(t.moneypaid) as s1
        from Passenger p left outer join(
                select *
                from Card natural join AddTime
                where year(attime) = '2018') as t on p.pid = t.pid
```

```
        group by p.pid;
create view v2 as
        select p.pid, sum(t.money) as s2
        from Passenger p left outer join(
                select *
                from Card natural join AddValue
                where year(avtime) = '2018') as t on p.pid = t.pid
        group by p.pid;
select pid, ifnull(s1, 0) + ifnull(s2, 0)
from v1 natural join v2;
```

pid	ifnull(s1, 0) + ifnull(s2, 0)
1	506
2	165
3	0
4	183
5	100
6	173
7	0

(iii)

```
select avg(cnt)
from (select count(c.cid) as cnt
        from Passenger p left outer join Card c on p.pid = c.pid
        group by p.pid) as t
```

avg(cnt)
1.7143

(iv)

```
select distinct pid
from Card natural join Ride
group by pid, date(rtime)
having count(*) > 10
```

pid
1

(v)

```
create view max_cnt as
        select count(cid) as cnt
        from Ride
        where year(rtime) = '2018'
        group by sid
        order by count(cid) desc
        limit 1;
select sname
from Station s natural join Ride r natural join max_cnt
where year(rtime) = '2018'
group by sid, cnt
having count(r.cid) = max_cnt.cnt;
```

sname
Jay St-MetroTech
Dekalb Av

(vi)

```
select distinct cid
from AddTime natural join Ride
where mday = 7 and cid not in (select cid
                               from AddTime natural join Ride
                               where mday = 7 and
                               rtime >= attime and
                               rtime < attime + interval '7' day)
```

cid
1
7
10

(g)

(i)

```
update Card
        set cexpireday = timestampadd(day, 1, cexpireday)
        where datediff(cexpireday, now()) >= 10
```

(ii)

```
delete from Card
        where cbalance = 0 and cexpireday < now()
```

(iii)

```
insert into AddTime
        select 1, now(), mday, mprice
        from TimePrice
        where mday = 30;
update Card
        set cexpireday = date_add(curdate(), INTERVAL 30 day)
```

Problem 2

(a)

```
create view ValueOnlyPassengerRide as
select pid, cid, rtime as ttime, rprice as tprice
from Card natural join Ride
where year(rtime) = 2018 and pid not in (
   select distinct pid
   from Card natural join AddTime
```

```
    where year(attime) = 2018)
```

(i)

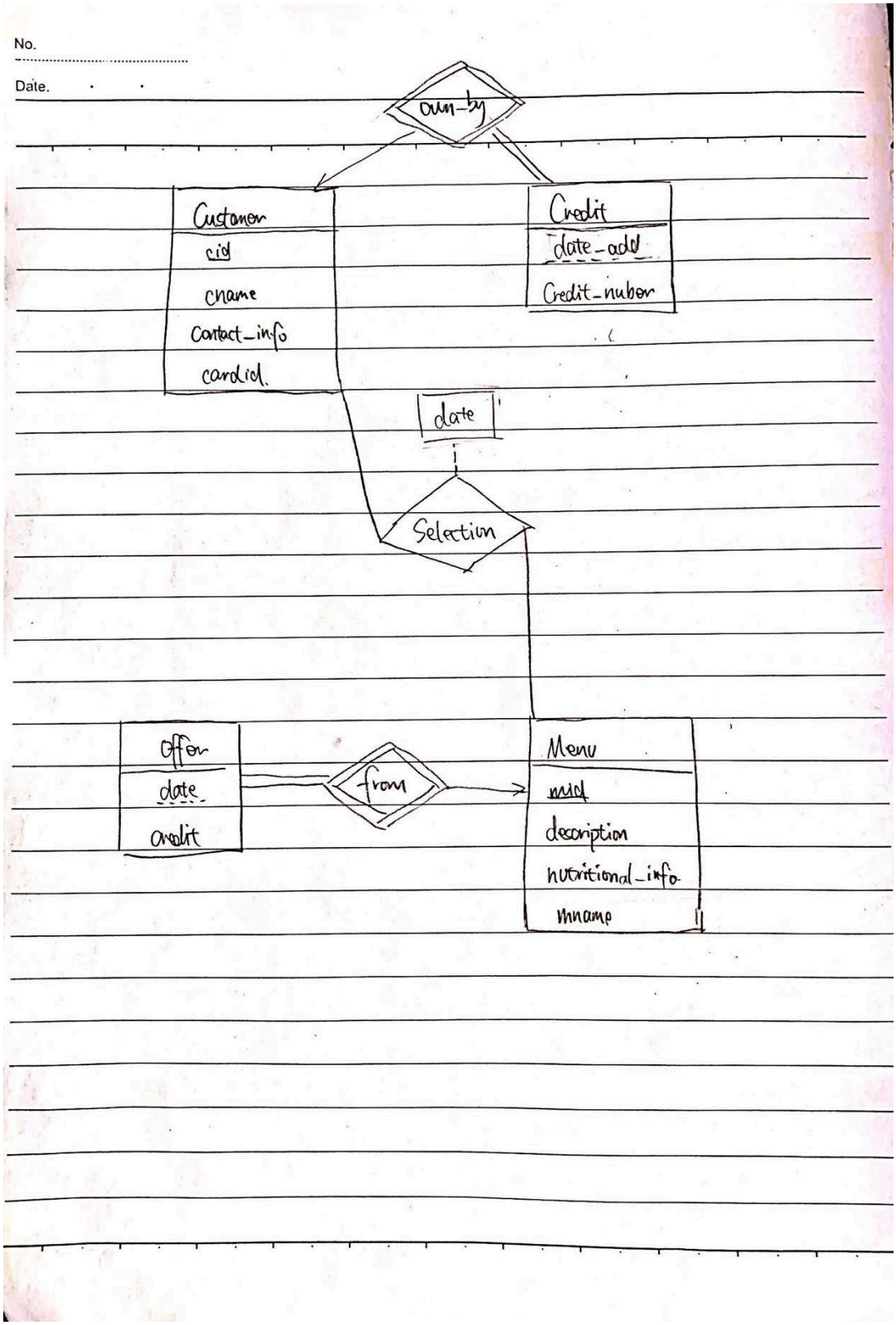
```
select pid
from
    (select mprice
    from TimePrice
    where mday = 30) as a
join
    (select pid, sum(tprice) as actualprice
    from ValueOnlyPassengerRide
    where month(ttime) = 6
    group by pid) as b
where mprice < actualprice
```

pid

(ii)

Problem 3

(a) Assumption : Every selection's mid is from the mid that day offer.


No.
Date.
own-by
Customer
cid
cname
Contact_info
cardid.
Credit
date_add
Credit_nubor
date
Selection
Offer
date
credit
from
Menu
mid
description
nutritional_info
mname

(b)

Customer (cid, cname, contact_info, cardid)

Credit (cid, date_add, credit_number) Foreign keys : cid referencing cid in Customer

Menu (mid, description, nutritional_info, mname)

Offer (mid, date, credit) Foreign keys : mid referencing mid in Menu

Selection (cid, mid, date) Foreign keys : cid referencing cid in Customer, mid referencing mid in Menu

(c) (i) Create temporary table t1

```
Select mid, count(distinct date) as count
From Offer
Where year(date) = "2018"
Group by mid;
Create temporary table t2
Select mid, ifnull(t1.count, 0) as times_offered
From Menu left outer join t1 on Menu.mid = t1.mid
Create temporary table t3
Select mid, count(distinct date) as count
From Selection
Where year(date) = "2018"
Group by mid;
Select t2. mid, times_offered, ifnull(t3.count, 0) as times_pickup
From t2 left outer join t3 on t2.mid = t3.mid
Drop table t1, t2, t3
```

(ii) Create temporary table t1

```
Select cid, month(date_add) as month_add, sum(credit_number) as credit
From Credit
Where year(date_add)  = "2018"
Group by cid, month_add
Create temporary table t2
Select cid, month(Selection.date) as select_month, sum(credit) as credit
```

From Selection, Offer

Where Selection.mid = Offer.mid and Selection.date = Offer.date and year(Selection.date) = “2018”

Group by cid, select_month;

Select t1.cid

FROM t1 left outer join t2 on t1.cid = t2.cid and t1.month_add = t2.select_month

Where t1.credit * 0.8 > t2.credit or t2.credit is null

Drop table t1, t2

(iii) Create temporary table t1

Select mid, date, count(*) as count

From Selection

Where year(date) = “2018”

Group by mid, date

Select t1_1.mid as midA, t1_2.mid as midB

From t1 as t1_1, t1 as t1_2

Where t1_1.mid != t2_2.mid and t1_1.date = t1_2.date and t1_2.count >= t1_1.count * 2

Drop table t1

Problem 4

1. For each student, output the number of events the student registered for in Fall 2023. (Hint: As there may be students attending 0 events, a natural join is not enough.

select s.sid, ifnull(numberofevents,0) **as** eventsAttendedCnt

from Student s left **outer join**

(

-- number of events (if more than 1) of every student in Fall 2023

select sid, count(eid) **as** numberofevents

from Register **natural join** Event **where**

edate > '2023-09-01 00:00:00'

and edate < '2024-01-01 00:00:00'

```
group by sid
) as t1
on t1.sid=s.sid
```

1. Output the name of the club that has the greatest increase in member fees in Spring 2024 compared with Fall 2023.

```
drop view if exists Fall23Fee;
drop view if exists Spring24Fee;
drop view if exists FeeIncrease;
create view Fall23Fee as
   (select distinct c.cid as cid,c.cname, ifnull(t1.memberfee,0) as fee
    from club c left outer join
         (select * from membership m
          where m.year = 2023 and m.semester="Fall") as t1
    on c.cid = t1.cid);
create view Spring24Fee as
   (select distinct c.cid as cid,c.cname, ifnull(t2.memberfee,0) as fee
     from club c left outer join
         (select * from membership m where m.year = 2024 and        m.semester="Spring") as t2
     on c.cid = t2.cid);

create view FeeIncrease as
  (select cid, Spring24Fee.fee - Fall23Fee.fee as increase
    from Fall23Fee join Spring24Fee using (cid));
select cid
from FeeIncrease
where increase = (select max(increase) from FeeIncrease);
```

3 Output the id, name, and date of the event co-organized by the largest number of clubs.

```
Create view NumClubs as
    (select eid, count(*) as counts
      from HoldsEvent
      group by eid) ;
select eid, ename, edate
from NumClubs natural join Event
Where counts = (select max(counts) from NumClubs)
```

4. Output the id and name of the events held in Fall 2023 that have a number of attendees equal to its maximum people allowance.

```
SELECT eid, ename
FROM Event NATURAL JOIN Register
WHERE edate > '2023-09-01 00:00:00'
and edate < '2024-01-01 00:00:00'
GROUP BY eid, ename, maxpeople
HAVING COUNT(*) = maxpeople
```

5. Fill in the blanks in following query to output the IDs and names of students (other than Bob) who belonged in Fall 2023 to all of the clubs that 'Bob' (sid 12345) belonged to that semester:

Method 1:

```
SELECT s.sid, s.sname FROM [Student AS s]
WHERE s.sid <> [12345]
AND NOT EXISTS
(
-- Bob's 2023 clubs EXCEPT this student's clubs
   SELECT * From [Membership]
   WHERE sid = 12345 AND semester = 'Fall' AND year  = 2023
```

```
   AND cid NOT in
-- Bob's F'23 clubs
        (SELECT [cid] FROM [Membership AS m2]
         WHERE [m2.sid] = s.sid AND semester = 'Fall' AND year = 2023))
```

Method 2:

```
SELECT s.sid, s.sname FROM Student s
WHERE s.sid <> 12345
AND
( -- number of clubs Bob is in during Fall 2023
SELECT COUNT(*)
  FROM Membership WHERE  [semester = 'Fall' AND year = 2023 AND sid = 12345])
=
(
-- number of clubs Bob and current student are both in during Fall 2023
SELECT COUNT(*)
 FROM Membership AS m1 JOIN Membership AS m2 USING (cid, semester, year)
WHERE semester = 'Fall' AND year = 2023 AND [m1.sid = 12345 AND m2.sid = s.sid])
```

6. Output the sid, sname, cid, cname, for each student and club such that the student has been a member of the club every semester since Fall 2023.

Full mark Version:

```
select sid, sname, cid, cname from
Membership natural join Student natural join Club
where
(semester = 'Fall' and year = 2023)
or
(semester = 'Spring' and year = 2024) group by sid,sname,cid,cname having count(*) = 2;
```

Bonus Version:

```
DROP VIEW IF EXISTS all_sem;
-- to make things easier, we assume such a view exists and someone will maintain it
-- with triggers or manually
CREATE VIEW all_sem(year, semester) AS
(SELECT 2023 AS year, 'Fall' AS semester
 UNION
 SELECT 2024, 'Spring'
 -- UNION
 -- SELECT 2024, 'Fall');
 );
SELECT cid, cname, sid, sname
FROM
   (SELECT * FROM Membership NATURAL JOIN Student NATURAL JOIN Club) AS t1
JOIN all_sem USING (year, semester)
GROUP BY cid, cname, sid, sname
having (count(*)=(select count(*) from all_sem));
```

Another Bonus Version using check for empty set difference:

Note that the alias here is the requirement of MySQL;

```
SELECT DISTINCT t1.cid, t1.cname, tn.sid, tn.sname
FROM
   Membership NATURAL JOIN Student as tn NATURAL JOIN Club as t1
WHERE NOT EXISTS (
-- the set of all years and semesters
-- except the years/semesters when current student belonged to current club should be empty;
-- similar idea to question 6
   SELECT * FROM all_sem AS t3
   WHERE (t3.year, t3.semester) NOT IN (
      SELECT t2.year, t2.semester
```

```
    FROM ( Membership as t2 NATURAL JOIN Student as t5 NATURAL JOIN Club t6)
    WHERE t1.cid = t6.cid AND tn.sid = t5.sid
  )
);
```

7. Output the id and name of the club that either hasn't held an event or held an event that no one registered for.

```
select cid, cname from Club
where
cid not in (select cid from HoldsEvent natural join Register);
```

8. Output the id and name of the club that has the highest average rating of events held in Fall 2023.

```
DROP VIEW IF EXISTS AvgRating;
CREATE VIEW AvgRating AS
( -- average rating of Fall 2023 events sponsored by each club
SELECT cid, AVG(rating) AS ar
  FROM Register NATURAL JOIN HoldsEvent NATURAL JOIN Event
  WHERE edate > '2023-09-01 00:00:00' and edate < '2024-01-01 00:00:00'
  GROUP BY cid);

SELECT cid, cname
FROM Club NATURAL JOIN AvgRating
WHERE ar = (SELECT MAX(ar) FROM AvgRating);
```

TRC: 5. Find the IDs and names of students (other than Bob) who belonged in Fall 2023 to all of the clubs that 'Bob' (sid 12345) belonged to that semester:

Main idea: find students s such that ((for all c, if c is a club that Bob belongs to in Fall 2023 then s belonged to c in Fall 2023) and s is not Bob). Since the only attribute we need about Clubs is the cid, we don't need to look at tuples from Club. Instead we can work directly with tuples from Membership.

$$\{t \mid \forall m \in Membership(m[sid] = 12345 \wedge m[semester] = \text{'Fall'} \wedge m[year] = 2023 \\ \rightarrow \exists m2 \in Membership(m2[sid] = t[sid] \wedge m2[cid] = m[cid] \wedge m[semester] = \text{'Fall'} \\ \wedge m[year] = 2023 \wedge m2[sid] \neq 12345)) \\ \wedge \exists s \in Student(s[sid] = t[sid] \wedge t[sname] = s[sname])\}$$